Introduction to Supervised Learning

Machine Learning

Daniele Loiacono

POLITECNICO
MILANO 1863

References

- This slides are based on material of prof. Marcello Restelli
- *Pattern Recognition and Machine Learning*, Bishop
  - Chapter 1

What is supervised learning? : problem of finding a good approximation of a function f that maps a given input to a target t

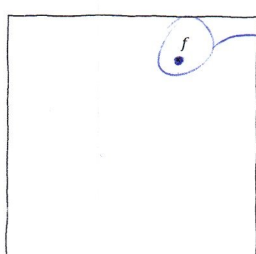
- It is the most popular and well established learning paradigm
- Data from an unknown function that maps an input x to an output t: $\mathcal{D} = \{\langle x, t \rangle\}$
- Goal: learn a good approximation of f
- Input variables x are usually called **features** or **attributes**
- Output variables t are also called **targets or labels**
- Tasks (depending on how this t is we call supervised learning in a different way):
  - **Classification** if t is discrete
  - **Regression** if t is continuous
  - **Probability estimation** if t is a probability

When to apply supervised learning?

- When **human cannot perform the task**
  - e.g., DNA analysis
- When human can perform the task but **cannot explain how**
  - e.g., medical image analysis
- When the task **changes over time**
  - e.g., stocks price prediction
- When the the task is **user-specific**
  - e.g., movie recommendation

Overview of Supervised Learning

- We want to **approximate** a function f given a data set $\mathcal{D}$
- The steps are



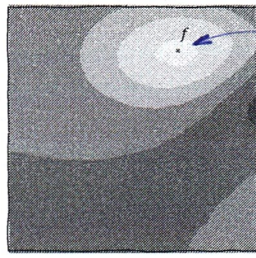
we believe that there exits the "true" function f

$\mathcal{F}$ = space of all the possible functions that can map the input domain into the target domain

Overview of Supervised Learning

- We want to **approximate** a function f given a data set $\mathcal{D}$
- The steps are
  - Define a **loss function** $\mathcal{L}$

function that tells us how good is the current approximation



$\mathcal{F}$

Suppose that this is the true function f that we want to approximate

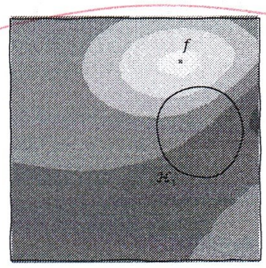
Suppose that this is the loss function: the darker regions represents approximations with bigger losses (worse approximations). The lighter the color, the better the approximation.

Overview of Supervised Learning

- We want to **approximate** a function f given a data set $\mathcal{D}$
- The steps are
  - Define a **loss function** $\mathcal{L}$
  - Choose the **hypothesis space** $\mathcal{H}$

we can't afford to look for a good approximation in an infinite space ⇒ we select an affordable subspace
(Note: affordable ≠ finite: if we choose a model which is a line we have ∞ lines but still only lines, nothing else)

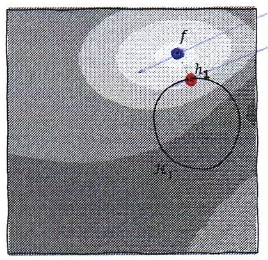
Family of models (of functions) that we'll inspect to find the best approximation



$\mathcal{F}$

Overview of Supervised Learning

- We want to **approximate** a function f given a data set $\mathcal{D}$
- The steps are
  - Define a **loss function** $\mathcal{L}$
  - Choose the **hypothesis space** $\mathcal{H}$
  - Find in $\mathcal{H}$ an approximation h of f that **minimizes** $\mathcal{L}$

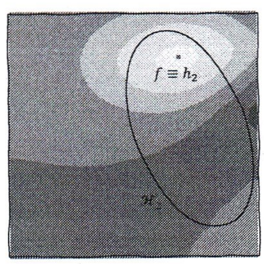


$\mathcal{F}$

This will be the non-reducible loss that we'll have

Overview of Supervised Learning

- We want to **approximate** a function f given a data set $\mathcal{D}$
- The steps are
  - Define a **loss function** $\mathcal{L}$
  - Choose the **hypothesis space** $\mathcal{H}$
  - Find in $\mathcal{H}$ an approximation h of f that **minimizes** $\mathcal{L}$
- What if we enlarge the hypothesis space?
  - We can approximate f without error!



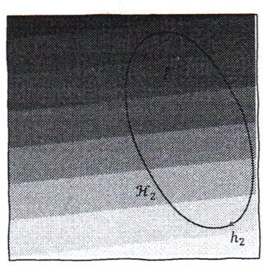
$\mathcal{F}$

If we enlarge the hypothesis space it can go very well, however we don't know the function f
⟹ we are not able to design a perfect loss function as in the figure (If we are able to design the perfect loss function then we already know the function we try to approximate → **NO SENSE**)

We approximate a loss function which may be like this

Overview of Supervised Learning

- We want to **approximate** a function f given a data set $\mathcal{D}$
- The steps are
  - Define a **loss function** $\mathcal{L}$
  - Choose the **hypothesis space** $\mathcal{H}$
  - Find in $\mathcal{H}$ an approximation h of f that **minimizes** $\mathcal{L}$
- What if we enlarge the hypothesis space?
  - We can approximate f without error!
  - But...



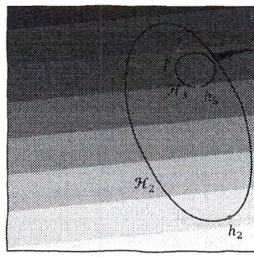
$\mathcal{F}$

(This is a little exagerated)

⟹ It's not always a good thing to enlarge the hypothesis space since we're not relying on the true (optimal) loss function

Overview of Supervised Learning

- We want to **approximate** a function f given a data set $\mathcal{D}$
- The steps are
  - Define a **loss function** $\mathcal{L}$
  - Choose the **hypothesis space** $\mathcal{H}$
  - Find in $\mathcal{H}$ an approximation h of f that **minimizes** $\mathcal{L}$
- What if we enlarge the hypothesis space?
  - We can approximate f without error!
  - But...



$\mathcal{F}$

In this case a smaller hypothesis space is more convenient.

Elements of Supervised Learning Algorithms

- Representation : how to choose the hypothesis space (H)
- Evaluation : loss function ($\mathcal{L}$)
- Optimization : search for the optimum given H and $\mathcal{L}$ (optimum : h)

How to represent the approximation function

How to evaluate the approximation

Find which one of the functions in the Hp. space is the one that minimizes the loss function

Examples of representation (H)

- **Linear models**
- **Instance-based**
- Decision trees
- Set of rules
- Graphical models
- Neural networks
- **Gaussian Processes**
- **Support vector machines**
- **Model ensembles**
- etc.

Examples of evaluation ($\mathcal{L}$)

- Accuracy
- Precision and recall
- Squared Error
- Likelihood
- Posterior probability
- Cost/Utility
- Margin
- Entropy
- KL divergence
- etc.

Examples of evaluation

- Accuracy
- Precision and recall
- Squared Error
- Likelihood
- Posterior probability
- Cost/Utility
- Margin
- Entropy
- KL divergence
- etc.

Examples of optimization

(h)

- Combinatorial optimization (if $|H|$ = finite (finite space))
  - e.g.: Greedy search
- Convex optimization
  - e.g.: Gradient descent
- Constrained optimization
  - e.g.: Linear programming

A Supervised Learning Taxonomy

- Parametric vs Nonparametric
  - Parametric: **fixed and finite** number of parameters

    we define the representation like a parametrized function and the set of parameters is fixed (these params are ⫫ of the data)
  - Nonparametric: the number of parameters **depends on thetraining set**

    it's not that there's no parameters, the parameters are not fixed, they depend on the training set
- Frequentist vs Bayesian
  - Frequentist: use probabilities to model the **sampling** process

    They leverage on probability but they don't model uncertainty (about the data and about the model)
  - Bayesian: use probability to **model uncertainty** about the estimate

    They use probability to model also uncertainty about the estimates
- Empirical Risk Minimization vs Structural Risk Minimization
  - Empirical Risk: Error over the **training set**

    How good we are at fitting the training data
  - Structural Risk: Balance training error with **model complexity**

    Balance between how good we are at fitting the data and also how complex the model is (the more complex the model → the more precise the performance on training data → the heigher the prob. of low performances on unseen data)
- Direct vs Generative vs Discriminative
  - Generative: Learns the **joint** probability distribution $p(x,t)$
  - Discriminative: Learns the **conditional** probability distribution $p(t|x)$

Direct, Discriminative, or Generative

- Our goal, is learn from **data** a **function** that maps **inputs** to **outputs**, t : targets (discrete/continuous)

$$\mathcal{D} = \{\langle x, t\rangle\} \Rightarrow t = f(x)$$

: we look for the function that we believe approximates the data ($f(\bullet)$)

- **Direct approach**
  - Learn directly an approximation of f from $\mathcal{D}$

  It does not leverage any probabilistic modeling. It directly tries to find an approximation of f that minimizes an objective function computed on the training set (we define a parametrized function and we optimize the parameters based on the data we collected)
- **Discriminative approach**
  - Model **conditional density** $p(t|x)$
  - Marginalize to find **conditional mean** $\mathbb{E}[t|x] = \int t \cdot p(t|x)\, dt$

  we model the problem with a probabilistic approach and we try to model the conditional density $p(t|x)$ = probability of getting some target given some input. Then we can marginalize and find the expected value. $\mathbb{E}[t|x]$ represents the most likely value of t given the input x.
- **Generative approach**
  - Model **joint density** $p(x,t)$
  - Infer **conditional density** $p(t|x)$
  - Marginalize to find **conditional mean** $\mathbb{E}[t|x] = \int t \cdot p(t|x)\, dt$

It models the joint probability density of the set. It's the probability of observing jointly the input and its target. This approach is far more complex than the previous, however it allows to many more stuff. Moreover, we can always derive $p(t|x)$ from $p(t,x)$ and obtain the discriminative approach. With $p(t,x)$ we can generate new samples (we cannot do it only with $p(t|x)$).